Human Face Detection using Injection Layer in YOLOv8n Model

Jelin Raphael Akkara
jelinraphael.akkara@studenti.unipd.it

Davide Checchia
davide.checchia@studenti.unipd.it

Kiamehr Javid
kiamehr.javid@studenti.unipd.it

Abstract

This study explores an innovative approach to enhance human detection, particularly small humans recognition, using a modified YOLOv8n model. We introduce a novel "Injection Layer" designed to emphasize blob keypoints, hypothesizing that this would improve small people detection capabilities. Our experiments involve various parameter adjustments, including image resolution and layer freezing techniques. We utilize the CrowdHuman Dataset for training and evaluation, employing metrics such as Precision, Recall, mAP50, and mAP50-95 to assess model performance. Our results show a consistent, albeit modest, improvement of around 2% in Recall detection performance with the Injection Layer. While the enhancement is smaller than initially anticipated, it demonstrates the potential of our approach. Our study provides valuable insights into YOLOv8n model optimization and paves the way for future enhancements in human detection algorithms.

1. Introduction

Human detection, particularly in crowded scenarios, remains a critical challenge in computer vision. Its applications span various domains, including surveillance, robotics, and autonomous vehicles. This study focuses on improving the YOLOv8n model, a state-of-the-art object detection algorithm for enhanced human detection with a particular emphasis on small humans recognition.

Our primary contribution is the introduction of an "Injection Layer," designed to preprocess images by emphasizing blob keypoints. We hypothesize that this emphasis on facial features leads to improvement in the model's ability to detect and recognize small persons. Additionally, we explore various model parameters and training strategies to optimize performance.

2. Related Work

Human detection has evolved from traditional computer vision techniques to advanced deep learning methods. Early approaches like the Histogram of Oriented Gradients (*HOG*) [1] laid the groundwork for pedestrian detection. The advent of Convolutional Neural Networks (*CNNs*) marked a significant leap, with architectures like R-CNN [2] and its successors improving both accuracy and speed.

The YOLO family [3] revolutionized object detection by framing it as a regression problem, enabling real-time performance. YOLOv8n, the latest iteration of the nano-models, represents the current state-of-the-art in balancing speed and accuracy. For crowded scenes, specialized approaches like multi-column CNNs [4] and CSRNet [5] have addressed challenges of occlusion and scale variation. Face detection, a subset of human detection, has progressed from cascaded classifiers [9] to deep learning methods like MTCNN [6].

Our work builds upon these foundations, while introducing novel elements like the Injection Layer to enhance feature detection in crowded scenes.

3. Dataset

Our study utilizes the CrowdHuman dataset[1], a comprehensive benchmark designed for evaluating human detection algorithms in crowded scenarios. CrowdHuman is notable for its large scale, rich annotations, and high diversity of human instances, making it ideal for our research on improving human detection in complex environments.

The complete dataset comprises 24,370 images (*15,000 for training, 4,370 for validation, and 5,000 for testing*), containing approximately 470,000 human instances across the training and validation subsets. On average, each image contains 23 persons, representing various occlusion scenarios, poses, and scales. CrowdHuman's annotation system is particularly detailed, with each human instance labeled using three types of bounding boxes: head, visible region, and

[1] https://www.crowdhuman.org/

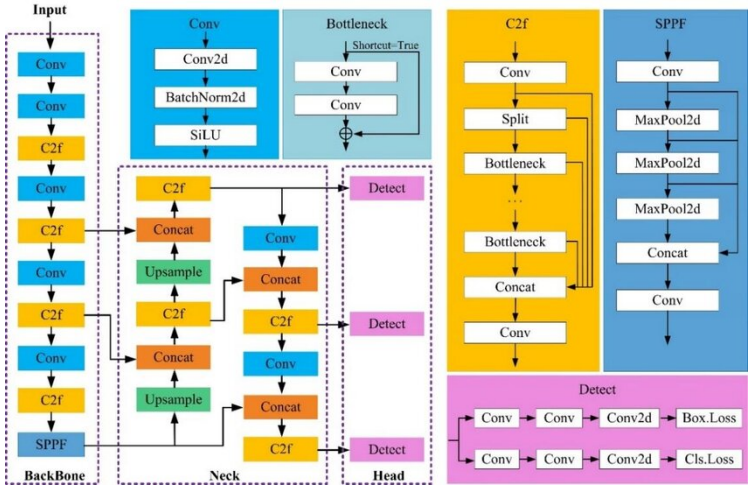


Figure 1: Architecture of the Yolov8n model.

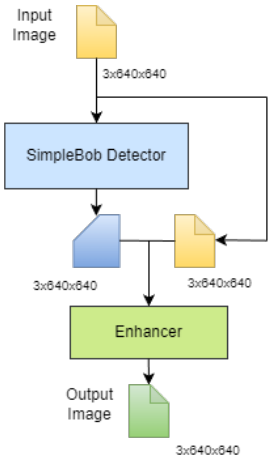


Figure 2: Diagram of the Injection Layer. Note that the Output of this layer goes directly into the YOLOv8n.

full-body. This multi-level annotation allows for nuanced evaluation of detection algorithms, especially in partially occluded scenarios.

For our study, we adapt the dataset to fit our research scope and computational constraints. We randomly select a subset of 1,500 images: 1,000 for training and 500 for validation.

3.1. Data Preprocessing

The original dataset comprises a vast array of human crowds, with each person carrying three different **annotations**. Each of these corresponds to the specifications (x_{bl}, y_{bl}, $width$, $height$)[2] of the head box, the visible body box and the full body box (*inferred from the image*).

For every picture present in our reduced dataset, the face and full body have been ignored, while the box pertaining the visible body is transformed into the format accepted by the YOLOv8 model (x_{center}, y_{center}, $width$, $height$).

3.2. Small Human Dataset

One of the major issues facing YOLOv8 is detecting far away persons. An improvement in this regards would mean we were able to better the model in one of its most challenging tasks. In order to do this, we carefully prepare a test dataset that contains images with only small person annotations. Obtaining a relatively higher recall (*detecting more small persons*) on this test data would correspond to the model progressing on this task.

In order to prepare the dataset, we set two thresholds. One, we set an area threshold, which is used to define the area for a small person annotation. Two, we set a count threshold, which counts the number of small person annotations. Using the latter threshold, we pick images that contain at least 10 small persons, as this is the metric we are testing the model on. We then delete all annotations above the area threshold, leaving us with a dataset containing small persons annotated images.

We repeat this procedure to generate two distinct test datasets of different sizes, namely 500 and 1,000 images.

4. Method

In our work we fine-tune a pre-trained version of the YOLOv8n Model and a custom made version, in which the Injection Layer is implemented.

4.1. YOLOv8 Model

YOLO (*You Only Look Once*) is a single-shot algorithm that efficiently detects objects in one pass. YOLOv8[3], the latest in the YOLO series, uses a single neural network to predict bounding boxes and class probabilities from the entire image input.

The architecture consists of three main components: backbone, neck, and head. The backbone, a pre-trained CNN, extracts multi-level feature maps from the input image. The neck, using components like the Feature Pyramid Network (*FPN*), integrates these feature maps. The head then processes the combined features for object classification and bounding box prediction.

Unlike two-stage detectors such as R-CNN, YOLO employs a one-stage dense prediction model in its head. Key

[2]Here *bl* stands for *bottom left* of the box.

[3]https://github.com/ultralytics/ultralytics

features of YOLOv8 include mosaic data augmentation, anchor-free detection, a C2f module, a decoupled head, and an optimized loss function, all contributing to its improved performance in object detection tasks.

The YOLO series consists, alongside different versioning, of diversely-sized models. In order to test the performance on our dataset, we employed the **nano** version of the model (*namely, **YOLOv8n**, package version 8.2.56*), of which an architectural diagram is present in Figure 1.

4.2. Injection Layer

Our key innovation is the introduction of an Injection Layer, designed to enhance small human recognition capabilities. The architecture of this layer is as follows:

1. The input image (*with variable resolution and standard RGB channels*) is processed through a Blob Detector.
2. The resulting blob-enhanced image is combined with the original input to produce an "enhanced" image, where blob-like features (*such as faces*) are more pronounced.
3. This enhanced image, maintaining the original input dimensions, is then fed into the standard YOLOv8 pipeline.

A brief diagram of the Injection Layer can be seen in Figure 2.

The Blob Detector and the Enhancer are, in our study, considered **deterministic**. We have explored different possibilities of layers present in the CV2 library[4], manually selecting both the most compatible with our face-detection goal and its settings.

4.3. Training Method

We employ several strategies to optimize our model:

- **Image Resolution:** We experimented with various input resolutions to balance detection accuracy and computational efficiency.
- **Number of Frozen Layers:** Due to the model complexity and overall presented accuracy, we vary the number of layers upon which the model is fine-tuned, using already-implemented options in the model definition.

Once the best configuration is found, the same parameters are then used in the model equipped with the Injection Layer.

It is important to note that every training is done setting the flag **single_cls** to **True**. This collapses every possible label into one single CLASS, defaulting any object that is not detected to NO_CLASS (*which is the background class for the YOLO family*). This setting allows the final layer of the model to focus on a single class (*in our case, human bodies*)

[4]https://opencv.org/

5. Experiments

5.1. Evaluation Metrics

Due to nature of identifying single elements across an image, YOLOv8 (*and, more importantly, YOLOv8n*) adopts a precursor to any metric in order to compare model-produced boxes to the ground truth, the **Intersection over Union** (*IoU*):

$$IoU(B_{pred}, B_{true}) = \frac{B_{pred} \cap B_{true}}{B_{pred} \cup B_{true}}$$

If a boolean value is required (*either the entity/object was identified or it was not*), the value is compared against a threshold. By default, this is set to 0.7 during inference activities, and 0.6 during validation ones.

This last operation allows us to employ standard metrics alongside model-dependent ones:

- **Precision**: The ratio of correctly predicted positive observations to the total predicted positive observations.
- **Recall**: The ratio of correctly predicted positive observations to all actual positive observations.
- **mAP50**: Mean Average Precision at 50% IoU.
- **mAP50-95**: Mean Average Precision over different IoU thresholds, ranging between 50% and 90% with a stepsize of 5%.

5.2. Injection Layer Implementation

To implement our novel Injection Layer, we modify the core YOLOv8n library. This process requires manually downloading the library and working on two separate branches:

- **Main Branch:** This branch maintains a standard implementation of YOLOv8n, serving as our control for comparison.
- **SimpleBlob Branch:** This branch contains our modified approach, where we hardcode the new Injection Layer directly into the library. This allows us to easily recall and utilize the custom layer in our Python code.

This dual-branch approach enabled us to directly compare the performance of our modified model against the standard YOLOv8n implementation, ensuring a fair evaluation of the Injection Layer's impact on small human detection tasks.

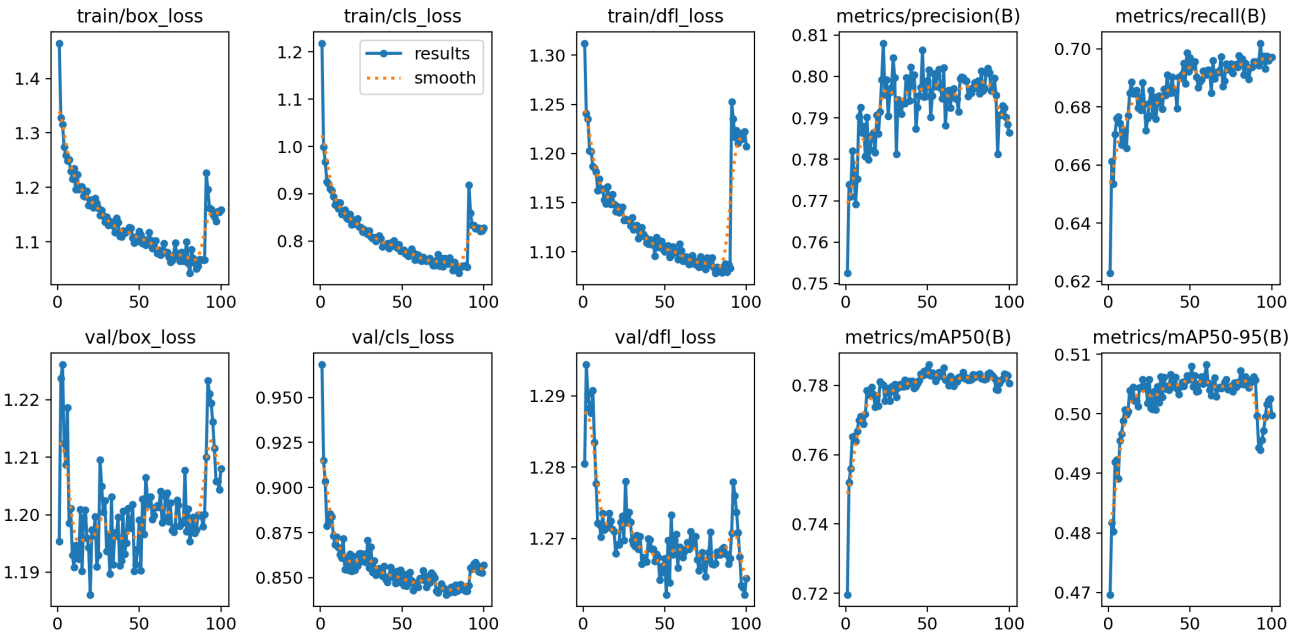


Figure 3: Results of the fine-tuning on the SimpleBlob branch model, using the best parameters inferred from the main branch model, on 100 epochs.

In order to choose the Blob Detector layer we apply different kinds of layers to few images of our dataset, picking the one who manages to capture the highest number of face-related features. An example of how the Blob Detector + Enhancer modify the original picture is found in Figure 4.

5.3. Parameter Tuning

We conducted a series of experiments to fine-tune the base YOLOv8 model on our dataset across 100 epochs, focusing on two key parameters:

- **Image Resolution:** We tested square images with resolutions of 640×640, 960×960, and 1280×1280 pixels. This range allowed us to evaluate the trade-off between detection accuracy and computational efficiency.
- **Number of Frozen Layers:** We experimented with freezing 10, 16, and 22 layers. These values were chosen strategically:
  1. *10 layers:* Freezes up to the backbone
  2. *16 layers:* Freezes up to half of the neck
  3. *22 layers:* Freezes up to complete neck (*only the head/detection layer remains*)

  The freezing action is cross-checked by comparing the previous and fine-tuned state dictionaries, which tells us which layer parameters have changed.

Table 1 presents the baseline values for the model, using Image size 640×640 (*default*) and fully frozen layers.

Table 2 presents the precision and recall values for each combination of these parameters. Our analysis suggests that a greater image resolution combined with the possibility of fine-tuning more layers in the model yields the optimal performance.

The best configuration is applied to the model in the SimpleBlob branch, yelding the training results in Figure 3. This model is then tested on the Small Human Dataset.

5.4. Small Human Detection

For the task of small human detection, we are majorly concerned with recall, which gives a good measure of how many of the short humans are being detected. Any large humans detected will only lower the precision, and not the recall, leaving the recall as a true measure of this task.

The validation was run both on the 500 and 1,000 images test small human datasets. We compare the performance of the YOLOv8n model using three different weights: pre-trained, fine-tuned (*without sharp blobs,* ***Main branch***), fine-tuned (*with sharp blobs,* ***SimpleBlob branch***). From Table 3, we see an increase in performance throughout, with the final model (*with injected sharp blobs*) giving the best performance of the three. This validates our hypothesis that finetuning with sharpened blobs increases the model performance, especially in a challenging task as detecting small humans.

Table 1: **(P)**recision, **(R)**ecall, **mAP50**, **mAP50-95** values for the pre-trained YOLOv8n model on the dataset (*baseline*).

P	R	mAP50	mAP50-95
0.725	0.522	0.626	0.394

Table 2: Precision and Recall after 100 epochs. Results are averaged across the last 10 epochs for each run.

Img Size	Frozen Layers		
	10	16	22
640	0.766, 0.606	0.748, 0.583	0.728, 0.524
960	0.783, 0.672	0.767, 0.655	0.757, 0.620
1280	**0.794, 0.693**	0.779, 0.678	0.763, 0.652

Table 3: Recall values on Small Human Dataset

Data Size	Model		
	Pre-train	Main	SimpleBlob
500	0.492	0.536	0.553
1000	0.489	0.553	0.564

An example of how the Injection layer operates on images can be found in Figure 5.

6. Conclusion

Our preliminary findings suggest that the Injection Layer (*and consequently, the emphasis on blob-like features*) may prove to be valuable in recognizing small humans in crowded images.

The intuitions regarding image resolution (*more resolution equating more useful details*) and the fine-tuning (*fewer frozen layers allowing for more information gathering*) remain as key findings.

Moving forward, we propose the following avenues for research:

- Refinement of the Injection Layer concept, possibly exploring alternative trainable approaches.
- Investigation of more sophisticated preprocessing techniques that could enhance detection without compromising the base model's performance.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

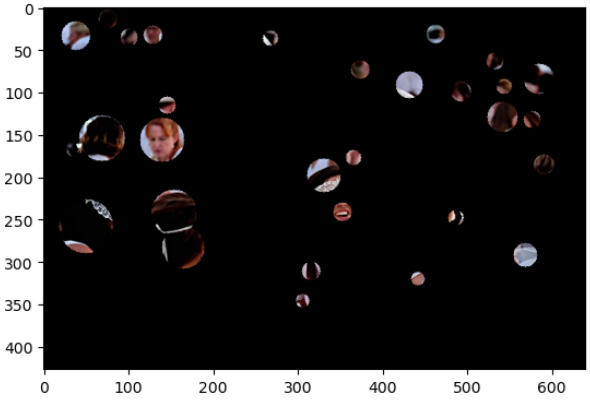
[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[5] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes, 2018.

[6] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, October 2016.

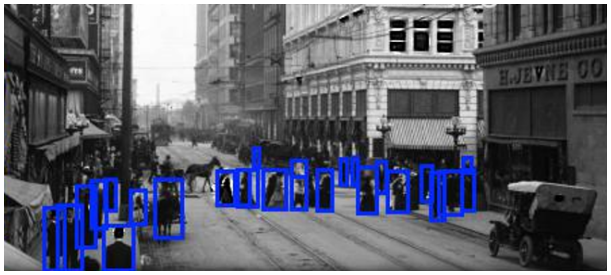
(a) Example of original picture from the dataset.



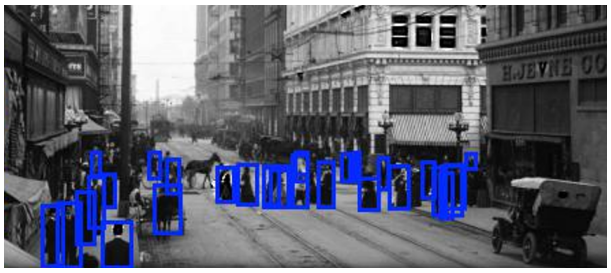
(b) Result of the Blob Detection layer applied to the original image.

(c) Result of the Injection Layer, before reaching the main YOLOv8n model. Notice the details corresponding to the blob detected in Figure 4b are more prominent.

Figure 4: Image processing in the Injection Layer pipeline.



(a) Example of an image evaluated on the Main branch (*no Blob detector has been applied*.



(b) Same example as Figure 5a evaluated on the SimpleBlob branch. Notice that more small people are correctly identified as opposed to the Main branch (*on the left*), whereas fewer, other classes are ignored, suggesting the validity of this approach.

Figure 5: Differences in evaluation of a single image, Main branch vs SimpleBlob branch.